UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

ANÁLISIS DE LA BIOMETRIA DE LAS RETINAS

Yihuan Gu

Tutor: David Domínguez Carreta

Mayo 2016

Análisis de la biometría de las retinas

AUTOR: Yihuan Gu

TUTOR: David Domínguez Carreta

Dpto. Grado en Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Mayo de 2016

Resumen (castellano)

Este documento está destinado a todos estudiantes y profesores que desean realizar una investigación hacia el análisis de la biometría, para ser más exacto, el proyecto esta referenciado a la memoria asociativa y los tipos que relevan de ella, como la memoria asociativa Hopfield. Mediante estos sistemas comenzaremos nuestra investigación.

Los capítulos de investigación pertenecerán al Capítulo 2, estado del arte, descripciones teóricas del funcionamiento de estos tipos de sistemas, algoritmos que se utilizan o utilizarán en nuestra investigación y las características de cada una de ella; Capítulo 3, arquitectura del sistema, análisis prácticos de los algoritmos a partir de los estudios teóricos del capítulo anterior, en esta sección se incluirán también análisis matemáticos y ejemplos de la forma de implementación del sistema, no se utilizará de ejemplo los cálculos realizados en nuestra investigación ya que esos patrones de entrada tienen una dimensión de 128x128=16 Megas de nodos y el tamaño de la matriz de peso de la fase de aprendizaje tendrán un tamaño de ⟦(128x128)⟧^2=256 Megas utilizaremos un vector más reducido para facilitar tanto al usuario como a nosotros de entender este sistema; Capítulo 4, resultados obtenidos a partir de nuestra investigación y desarrollo implementada.

En la parte de anexo se incluye algunas funciones, en lenguaje C, utilizadas en el sistema.

Palabras clave (castellano)

Memoria, patrones, entrenamiento y test, clasificación de patrones, reconocimiento de patrones, memoria, recuperación, conjunto fundamental, patrón fundamental, función de activación, retinas, umbral…

Abstract (English)

This report is intended for all students and teachers who wish to conduct research into artificial neural networks, to be more precise, the project is referenced to the associative memory and the types relieving it as the Hopfield associative memory. Through these systems begin our investigation.

Research chapters belong to Chapter 2, state of the art, theoretical descriptions of how these types of systems, algorithms used or used in our research and characteristics of each of them; Chapter 3, system architecture, practical analysis of algorithms from theoretical studies of the previous chapter, this section also mathematical analysis and examples of how to implement the system is included, will not be used for such calculations in our research since these input patterns have a dimension of 128x128 = 16 MB of nodes and the size of the weight matrix of the learning phase have a size 〖(128x128)〗 ^ 2 = 256 Megas we use a smaller vector for facilitate both the user and us to understand the system; Chapter 4, results from our research and development implemented.

In Part annex it includes some functions in C language used in the system.

Keywords (inglés)

Memory, patterns, training and test, pattern classification, pattern recognition, memory, recovery, fundamental set, fundamental pattern, activation function, retinas, threshold...

Agradecimientos

Ante todo, quiero expresar mi profundo agradecimiento a David Domínguez, mi tutor de la investigación, por su paciencia y su tiempo dedicado en la ayuda de la elaboración de esta memoria y sus conocimientos hacia la investigación de la inteligencia artificial y las redes neuronales. También se deben de agradecer a todos los profesores académicos que tuve a lo largo de la carrera en las enseñanzas proporcionadas.

Contenido

1 Introducción

1.1 Motivación

Los pequeños avances en la tecnología que se presentan actualmente han dado una diferencia exponencial respecto a los primeros desarrollos de la informática del siglo XX, el simple almacenamiento de la información digital ya no puede abastecer a la ambición del hombre hacia sus nuevas investigaciones y desarrollos. Este avance hace la necesidad de aumentar el uso de dispositivos con un mayor rendimiento y mayor seguridad. Para conseguir ambos objetivos hemos optado por la investigación de este trabajo, ya que incluye un conocimiento sobre la tecnología de información y la seguridad de sistemas informáticos.

La tecnología de la información almacenada en memoria es procesada por unidades centrales de procesamiento mediante dispositivos de entrada/salida. Toda esa información es guardada durante un intervalo de tiempo con un tamaño limitado, ya que todos los dispositivos de almacenamiento tienen una duración y una capacidad limitada. La extracción de información puede ser a veces muy difícil por los diferentes métodos de almacenamiento, que estos también pueden suponer un alto coste en tiempo y dinero para el usuario. Para ello, se desean optimizar el rendimiento de forma más inteligente y barata, con lo que separaremos la investigación en dos fases: fase de aprendizaje automático, algoritmos de agrupación de los diferentes tipos de datos; y la fase de recuperación, proceso de búsqueda que direcciona al resultado más probable mediante los cálculos matemáticos de estadística.

En este punto también queremos hacer una pequeña mención sobre la seguridad informática[1], ya que nos pareció interesante por el tema de la biometría que es el tipo de sistema que actualmente más se utiliza para el acceso a una información de contenido personal o privado, como el uso de los sensores de la huella dactilar, acceso por voz, comparación del rostro facial, etc. Pero esta área de investigación no será el tema que analizaremos, simplemente hacemos un hincapié la importación de los diferentes algoritmos de la biometría.

Por todas esas razones anteriores nos hemos propuesto a poner un pequeño granito de arena a la investigación de los diferentes algoritmos de la biometría de la retina con su aprendizaje automático de la información para su posterior recuperación. Los patrones que utilizaremos para ello serán imágenes de retinas visualizadas desde diferentes ángulos, evaluaremos la biometría de todos los patrones de nuestra minería de datos y se comenzará la recuperación de un nuevo patrón con ruido.

1.2 Objetivos

La preparación de esta memoria estará caracterizada por un diario general que recompilará todas las consultas, análisis, desarrollos y pruebas que se realizarán en las diferentes fases de la investigación. El objetivo del presente trabajo será la recuperación de patrones de retinas distorsionadas por diferentes motivos, que estos podrían ser como: por acción

[1] *La seguridad de tecnologías de la información o ciberseguridad, es el área de la informática que se hace responsable de la protección de la infraestructura computacional y todo lo relacionado a ella y, especialmente, la información contenida y circulante.* (Wikipedia)

humana de forma intencionada, para las pruebas que se realizan sobre el análisis matemático del programa desarrollado o robo de información y destrucción de ello por usuarios maliciosos; por la reconstrucción de patrones que han sido alterados por elementos externos como perdida de información cuando se realiza una importación o exportación de datos o desconocimiento de información de dicha parte del patrón; o simplemente podríamos utilizar un patrón original que nos sirva para observar el análisis que realiza en la misma recuperación y la diferencia que existe entre el nuevo patrón recuperado respecto al patrón original.

El éxito de una recuperación no está en visualizar el resultado final obtenido por el programa, sino en fijarnos en cada uno de los pasos de cálculo realizados para llegar a ella. Se deberá prestar importancia a la etapa de aprendizaje automático, ya que su resultado podría afectar en relevancia al nuevo patrón recuperado. Para ello, en primer lugar, será necesario seleccionar y evaluar los diferentes patrones de entrada que serán utilizados para el cálculo. Los patrones de aprendizaje serán obtenidos a partir de resultados obtenidos por otros investigadores por lo que nuestra investigación simplemente estará destinada al análisis de aprendizaje automático, recuperación de nuevos patrones de entrada y un aprendizaje automático continuo.

Cabe mencionar que para este tipo de estudio utilizaremos la Inteligencia artificial para las diferentes fases del proceso. Estos patrones serán tratados con las teorías de redes neuronales de conexión con las diferentes entradas de patrones de training and test.

1.3 Organización de la memoria

La memoria constará de los siguientes capítulos:

- Capítulo I: **Introducción**. Descripción resumida de las motivaciones y causas que condujeron a los estudiantes de grado en ingeniería informática a la investigación de este trabajo y los objetivos que cumplirán para llegar de forma exitosa al final de su investigación.
- Capítulo II: **Estado del arte**. Historias y definiciones sobre el uso de la biometría. Introducción a los patrones de redes neuronales y el uso de la memoria asociativa. Tipos de memoria asociativa y teoremas en la que se ha basado la red de Hopfield. Tipo de algoritmo que utilizaremos en nuestra investigación y la comparación de los diferentes algoritmos activación para el aprendizaje automático.
- Capítulo III: **Diseño y desarrollo**. Consultas y análisis teóricas de los diferentes algoritmos matemáticos y algebraicos que se podrían utilizar para obtener los resultados más óptimos, en probabilidad de error, y de menor coste, en número de cálculos. Traducción del lenguaje matemático a lenguaje máquina y optimizar los algoritmos de cálculos en función de números de instrucciones a ejecutar, número de veces que se accede a la memoria, memoria ocupada y tiempo de ejecución.
- Capítulo IV: **Integración, pruebas y resultados.** Análisis de los resultados obtenidos en comparación del diseño teórico del desarrollo práctico. E imágenes de los resultados obtenidos en comparación de los patrones de entrada original.
- Capítulo V: **Conclusiones y trabajo futuro.** Descripción del trabajo realizado, así como los resultados obtenidos, partes positivas de los investigadores del trabajo, partes que se deberán de mejorar y de tener en cuenta para los demás futuros posibles investigadores.

- Capítulo VI: **Referencias.** Diferentes fuentes bibliográficas utilizadas por los estudiantes para la investigación de esta.
- Apartado I: **Glosario.** Diccionario de siglas y acrónimos utilizados en la memoria.
- Apartado II: **Anexos.** Otros datos de interés utilizados en la investigación.

2 Estado del arte

2.1 Definición e historia

Según la Real Academia Española la palabra ***biometría*** está definida como un *estudio mensurativo o estadístico de los fenómenos o procesos biológicos*, es decir, es todo aquel que identifica características propias de un ser humano.

Desde hace mucho tiempo, la humanidad ha comenzado a utilizar características fisiológicas para la identificación y autentificación de registros personales. Estos actos se comenzaron a utilizarse en documentos escritos donde las personas marcaban con huellas de mano o firmas que simbolizaba la forma de no repudio. Los primeros documentos que se firmaban con huellas dactilares podrían figurarse en los años 500 a.C. Según las anotaciones históricas occidentales, el explorador y escritor *Joao de Barros*[2] describió que este tipo de sistema ya era utilizado en China, desde al menos el siglo XIV, donde los comerciales chinos lo utilizaban para hacer marcas de hulla de las manos de los niños esclavos para su identificación. En el mundo occidental esta práctica de reconocimiento no fue comenzada hasta finales del siglo XIX, donde las primeras personas que se "beneficiaron" de su uso fueron la policía francesa para la identificación de los criminales por las huellas dactilares en el año 1883.

Desde el punto de la tecnología de la información podemos considerar la biometría como un estudio que analiza el reconocimiento de las características físicas e intransferibles de una persona, por ejemplo, el ADN (*ácido desoxirribonucleico*). Actualmente, el análisis de la biometría es llevada a cabo mediante estudios científicos, herramientas informáticas y análisis matemáticos, como la ciencia audiovisual, sensor biológico y datos de la estadística biológica. El trabajo de este documento se centrará en un estudio que evaluará los datos estadísticos de la biometría de las retinas donde dichos valores numéricos han sido obtenidos a partir de la transformación de los patrones a números.

2.2 La biometría

En estos últimos años, el uso de sistemas de identificación con sensores biométricos cada vez se hace más popular, esto se debe a la simplicidad, la comodidad y la seguridad que se proporcionan y las garantías de autenticidad e integridad que obtenemos en los resultados. A diferencia de los métodos tradicionales que se utilizan como herramienta de acceso, como la llave de la puerta, tarjeta de identidad, clave de acceso, etc., estos poseen la desventaja de que pueden ser extraviados con facilidad y ser utilizado por un tercero malicioso. La ventaja que posee la biometría, respecto a los factores anteriores, está en que éste pertenece al propio individuo, es decir, pertenece a una característica física e intransferible de las personas, lo que le hace que sea más seguro y más práctico, eliminando casos de un posible extravío, y aumentando la dificultad de copias y falsificaciones sin consentimiento del propietario.

Pregunta: *"¿Qué pasaría si consiguen robarnos una identidad de la biometría?"*

[2] *João de Barros (1496 - 20 de octubre de 1570), apodado el Tito Livio portugués, puede decirse que fue el primer gran historiador de Portugal.* (Wikipedia)

Actualmente existen múltiples tipos de biometría que se utilizan para la verificación de la identidad, como el sistema biométrico del iris del ojo, la retina del ojo, huellas dactilares, geometría de la mano, rosto de la cara, análisis de voz y escritura y firma, existiendo a parte de ellas muchos más. Cada sistema de biometría ha tenido un grado de evolución diferente debido a los distintos métodos de extracción de las muestras, por ejemplo, para la extracción de una muestra de la huella dactilar simplemente tenemos que acercar el dedo a un sensor láser para que lo escanee, sin embargo, para obtener una muestra del iris del ojo tendríamos que acercar el globo ocular a un sensor láser. Con los ejemplos anteriores y recopilando información de estudios antecedentes adjuntamos una tabla comparativa que evalúan el nivel de uso, seguridad y aceptación pública que obtenemos en cada una de ellas:

	Ojo (Iris)	*Ojo (Retina)*	*Huellas dactilares*	*Geo. de la mano*	*Escritura y firma*	*Voz*
Fiabilidad	*Muy alta*	*Muy alta*	*Muy alta*	*Alta*	*Media*	*Alta*
Facilidad de uso	*Media*	*Baja*	*Alta*	*Alta*	*Alta*	*Alta*
Prev. de ataques	*Muy alta*	*Muy alta*	*Alta*	*Alta*	*Media*	*Media*
Aceptación	*Media*	*Baja*	*Alta*	*Alta*	*Muy alta*	*Alta*
Estabilidad	*Alta*	*Alta*	*Alta*	*Media*	*Baja*	*Media*
Nota media (sobre 5)	**3,75**	**3,25**	**4,00**	**3,50**	**3,00**	**3,25**

Tabla 1: Tabla comparativa de sistemas biométricos.

En la tabla anterior hemos evaluado de forma numérica los diferentes grados de aceptación, donde podemos encontrar el mejor y el peor sistema de biometría entre los evaluados, siendo en estos casos, la biometría de las huellas dactilares y la biometría de escritura y firma, respectivamente. No es difícil de entender los resultados obtenidos, la escritura y firma ha obtenido como el peor resultado ya que la calidad de la caligrafía varia en el tiempo dependiendo de su práctica, y como "el mejor sistema", las huellas dactilares, este método ha resultado como el más exitoso por su simplicidad en la hora de obtener una muestra, los valores obtenidos tienen permanencia, es decir, no cambiar de patrón después de los años. Aunque la biometría de la retina, tema sobre el que realizaremos la investigación, no ha resultado como unos de los mejores sistemas, podemos observar en la tabla que los valores obtenidos en la fiabilidad, estabilidad y prevención de ataque ha sido de los mejores, este sistema no ha tenido tanta popularidad como el de las huellas dactilares por su complejidad en la hora del uso y la poca aceptación del público hacia ella.

2.2.1 La retina

La **retina** *es una capa de tejido sensible a la luz que se encuentra en la parte posterior del globo ocular*. Está formada básicamente por distintas capas de neuronas interconectadas

mediante sinapsis[3]. Este componente del ojo ocular tiene como función la transmisión de señales eléctricas mediante nervios ópticos al cerebro.

2.2.2 Análisis de las retinas

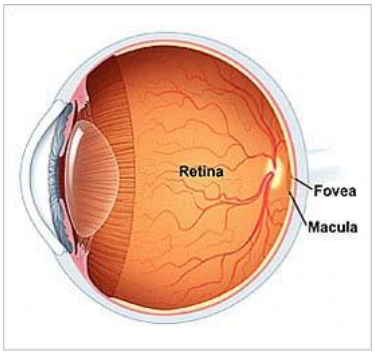


Imagen 1: La retina

El estudio de los patrones de la retina tuvo un origen en los años 30, donde los científicos biólogos establecieron la teoría de que la disposición de los vasos sanguíneos era diferente en cada persona, por lo que se da la propiedad de unicidad. Pero este concepto no fue totalmente popularizado hasta el año 1978 donde se patentó el primer escáner de la retina. Actualmente se realiza el estudio de los patrones mediante la distribución de los vasos sanguíneos que procede del nervio óptico y aparecen distribuidos en la retina. Este sistema de biometría cumple con la propiedad de singularidad y permanencia, es decir, la distribución aleatoria de los vasos sanguíneos hace que sea altamente diferente un patrón de otro y que es permanente a lo largo de la vida ya que su posición no variará.

El escaneo de las retinas requiere el uso de lectores con una fuente de luz infrarroja de baja intensidad para reflejar el patrón único de la retina utilizado para la identificación del individuo, este nuevo patrón obtenido será uno de los patrones de entrada que utilizaremos en los algoritmos de aprendizaje.

2.3 Memoria asociativa

Antes de empezar con el análisis de la biometría comenzaremos conociendo que es la memoria asociativa. Describimos la memoria asociativa como un sistema de aprendizaje automático que almacena y recupera información a partir de datos parciales aprendidos anteriormente. A veces también es denominada como la memoria de direccionamiento por contenido.

La memoria asociativa posee una estructura o algoritmo de funcionamiento similar al cerebro humano, poniendo un ejemplo de nuestra vida cotidiana, leer una novela. Cuando nosotros leemos un libro no tenemos guardado en nuestro cerebro la página y el capítulo exacto de cada palabra, sino que almacenamos los contenidos que consideramos más importantes como el nombre de personajes, lugar de la historia y sucesos que han ocurrido, si en un momento posterior a la lectura se nos pasa por la cabeza un fragmento de la novela, automáticamente lo relacionaremos con el nombre de los personajes y el lugar que ocurrió la historia. La investigación de este documento se hace principalmente sobre este tipo de memoria, como comentábamos en la introducción, la capacidad de almacenamiento es muy limitada, por lo que no tiene sentido que almacenáramos todos los patrones de la biometría, sino solo las partes que nos interesan.

[3] **Sinapsis** (RAE): *Conexión entre el axón de una neurona y la dendrita de otra cercana mediante neurotransmisores.*

En esta investigación utilizaremos la memoria asociativa como una herramienta para la recuperación de patrones completos a partir de patrones de entrada que pueden estar alterados con ruido aditivo, sustractivo o combinado. En base a esta afirmación, una memoria asociativa **w** puede formularse como un sistema de entrada y salida:

$$Patrón\ entrada\ (x) \rightarrow \text{Matriz aprendizaje(w)} \rightarrow \text{Patrón}\ salida(y)$$

Para simplificar de la representación de los patrones denominaremos como patrón de entrada como un vector con el símbolo **x** y el patrón de salida, un vector **y**. Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida. La memoria asociativa **w** se representa con una matriz cuya componente ij-ésima es $\boldsymbol{w_{ij}}$. La nueva matriz que se genera es obtenida a partir de un conjunto finito de patrones de entrada. Utilizando el símbolo μ como un índice, el conjunto de patrones de entrada y salida se representaría de la siguiente forma: $\{(x^{\mu}, y^{\mu}) \mid \mu = 1,2, \dots, p\}$.

Dependiendo de los valores de los patrones de entrada y salida podemos clasificar la memoria asociativa en dos tipos:

- Memoria auto-asociativa: $x^{\mu} = y^{\mu}\ \forall \mu\ \in \{1,2, \dots, \text{p}\}$.
- Memoria hetero-asociativa: $x^{\mu} \neq y^{\mu}\ \exists \mu\ \in \{1,2, \dots, \text{p}\}$

Los patrones fundamentales (patrones de entrada) que utilizaremos en nuestra investigación pueden estar alterados por diferentes tipos de ruidos, para diferenciar los patrones alterados de los patrones fundamentales utilizaremos el símbolo de la prima, siendo a partir de ahora x' la representación del patrón alterado. Como no tenemos certeza de que los patrones de salida mantienen su valor sin alteraciones, representaremos también con el símbolo y' como el patrón de salida alterado.

La dimensión de los vectores de los patrones de entrada y salida tienen un tamaño diferente. Sean **m**, **n** números enteros positivos. Se asigna como **m** la dimensión de los patrones de entrada, y por **n** la dimensión de los patrones de salida. La dimensión del patrón de entrada no tiene por qué ser igual a la dimensión del patrón de salida, aunque para nuestra investigación estos dos números enteros si serán iguales. Uno de los requisitos que debe cumplir una memoria auto-asociativa es que la dimensión de los patrones de entrada sea igual a la dimensión de los patrones de salida; por otro lado, si en una memoria sucede que m ≠ n, es evidente que la memoria debe ser hetero-asociativa.

2.3.1 Teoría hebbiana

La teoría hebbiana, también conocida como el aprendizaje de coincidencia, fue introducida por Donald Hebb en el año 1949. Su investigación fue realizada a partir de las redes neuronales biológicas donde describía que una sinapsis era reforzada si la neurona de entrada era continua a la neurona de salida, es decir, si ambas neuronas estaban conectadas entre sí. De las investigaciones existentes sobre la teoría hebbiana se han extraído la siguiente afirmación:

"Supongamos que la persistencia de una actividad repetitiva (o "señal") tiende a inducir cambios celulares duraderos que promueven su estabilidad. ... Cuando el axón de una célula A está lo suficientemente cerca como para excitar a una célula B y repetidamente toma parte en la activación, ocurren procesos de crecimiento o cambios metabólicos en una o ambas células de manera que tanto la eficiencia de la célula A, como la capacidad de excitación de la célula B son aumentadas."

Podemos resumir la teoría hebbiana con la siguiente ecuación: $w_{ij} = x_i x_j$, donde w_{ij} es la sinapsis de las neuronas, x_i y x_j son las dos neuronas interconectadas. No hay necesidad de saber cuál es la neurona de entrada y cuál de salida, ya que el peso de la conexión de las dos neuronas tiene el mismo valor indiferentemente de la dirección de la conexión.

2.3.2 Red de Hopfield

La red de Hopfield o modelo de Hopfield es una red neuronal artificial investigada por el señor John Hopfield, donde fue publicada en el libro *"Las redes neuronales y sistemas físicos con habilidades computacionales colectivos emergentes"* en el año 1982. Esta red también es denominada como la memoria asociativa de Hopfield, ya que se caracteriza en la utilización del sistema de memoria asociativa de aprendizaje y recuperación con unidades binarias. La principal característica de las unidades binarias es que tanto las entradas de los patrones como sus salidas actúan únicamente con dos posibles valores, en un principio propuso Hopfield valores para las neuronas de $x_i \in \{0, 1\}$, estado similar a la que propuso **McCulloch-Pitts**[4], sin embargo, en una sección de artículo posterior, *Studies of the collective behaviors of the model*, hicieron la relevancia de que el nivel de recuperación de los patrones en la capacidad de almacenamiento de la información se puede incrementar por un factor de dos, por lo que también se escogieron como posibles valores $x_i \in \{-1, 1\}$.

Hemos escogido este sistema como el principal algoritmo de aprendizaje y recuperación para nuestra investigación ya que se adapta perfectamente a las imágenes de retinas de mapeo blanco y negro, a partir de ahora describiremos de forma matemática con el valor positivo *uno*, el color *blanco* y con el valor negativo *menos uno*, el color negro.

2.3.2.1 Fase de aprendizaje

El modelo de Hopfield está basada en la teoría hebbiana que describe un método básico de interconexión sináptica que se incrementa si las neuronas tienen una conexión directa con todas las demás neuronas del patrón. La estructura resultante del aprendizaje hebbiana es una matriz de peso cuadrática y simétrica, es decir, el valor del peso de la conexión entre dos neuronas es igual es ambas direcciones y el valor de conexión consigo misma es nula.

$$w_{ij} = \begin{Bmatrix} \sum_{\mu=1}^{p} x_i^{\mu} x_j^{\mu} & si\ i \neq j \\ 0 & si\ i = j \end{Bmatrix}$$

Ecuación 2-1: Algoritmo de aprendizaje

Operativamente, el resultado de la expresión del Algoritmo de aprendizaje (***Ecuación 1***) se obtiene en tres pasos:

1. Para cada conexión del patrón fundamental se encuentra la matriz $x^{\mu}(x^{\mu})^t$ de dimensiones de n*n:

[4] McCulloch Pitts neurona: Definición basada en la neurona formal del cerebro humano y animal en el año 1943. Tuvo su importancia en los estudios matemáticos posteriores en la implementación lógica, una de las características que se concluyeron es que *"La actividad neuronal es un proceso de todo o nada"*.

$$x^{\mu}(x^{\mu})^{t} = \begin{pmatrix} x_1^{\mu} & x_2^{\mu} & x_n^{\mu} \end{pmatrix} \begin{pmatrix} x_1^{\mu} \\ x_2^{\mu} \\ x_n^{\mu} \end{pmatrix} = \begin{pmatrix} x_1^{\mu}x_1^{\mu} & x_1^{\mu}x_2^{\mu} & x_1^{\mu}x_n^{\mu} \\ x_2^{\mu}x_1^{\mu} & x_2^{\mu}x_2^{\mu} & x_2^{\mu}x_n^{\mu} \\ x_n^{\mu}x_1^{\mu} & x_n^{\mu}x_2^{\mu} & x_n^{\mu}x_n^{\mu} \end{pmatrix}$$

Ecuación 2-2: Cálculo 1 de aprendizaje

2. La matriz resultante es restada por la matriz identidad I[5], dejando el resultado con la diagonal principal a 0´s de tamaño n*n.

$$x^{\mu}(x^{\mu})^{t} - \mathrm{I} = \begin{pmatrix} \mathrm{x}_1^{\mu} & \mathrm{x}_2^{\mu} & \mathrm{x}_n^{\mu} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1^{\mu} \\ \mathrm{x}_2^{\mu} \\ \mathrm{x}_n^{\mu} \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & \mathrm{x}_1^{\mu}\mathrm{x}_2^{\mu} & \mathrm{x}_1^{\mu}\mathrm{x}_n^{\mu} \\ \mathrm{x}_2^{\mu}\mathrm{x}_1^{\mu} & 0 & \mathrm{x}_2^{\mu}\mathrm{x}_n^{\mu} \\ \mathrm{x}_n^{\mu}\mathrm{x}_1^{\mu} & \mathrm{x}_n^{\mu}\mathrm{x}_2^{\mu} & 0 \end{pmatrix}$$

Ecuación 2-3: Cálculo 2 de aprendizaje

3. La matriz peso o matriz de la memoria asociativa es acumulativo, es decir, el peso w_{ij} es igual al sumatorio de las conexiones de todos los patrones fundamentales.

$$w = \sum_{\mu=1}^{p} [x^{\mu}(x^{\mu})^{t} - I] = \left[W_{ij}\right]_{n*n}$$

Ecuación 2-4: Cálculo 3 de aprendizaje

2.3.2.2 Fase de recuperación

A partir de la matriz de peso *w* de las conexiones calculadas proseguiremos la fase de recuperación de los patrones. La fase de recuperación mediante el método de Hopfield consiste en *"recordar"* los posibles valores aprendidos en la fase anterior. Partiendo de la matriz de peso se calculará el posible valor de cada neurona del patrón distorsionado $x´$ mediante un valor umbral θ, para que el patrón de salida tenga también valores binarios, siendo la formula de recuperación de la siguiente manera:

$$x´_i = \sum_{j=1}^{n} w_{ij}\, x_j - \theta_i$$

Ecuación 2-5: Algoritmo de recuperación

Para mejorar la recuperación de los patrones distorsionados estableceremos estados en la memoria Hopfield en el patrón x´(t) del tiempo t, y en el tiempo t+1 siguiente patrón x´(t+1). Dado un vector de patrón fundamental de entrada x´, la fase de recuperación, igualmente que la fase de aprendizaje, consta de tres pasos:

1. Para el tiempo inicial t = 0, se hace $x´_i(0) = x´_i \; \forall \mathrm{i} \in \{1,2,..,\mathrm{n}\}$.

[5] La matriz identidad es una matriz que cumple la propiedad de ser un elemento neutro del producto de matrices.

2. $\forall i \in \{1,2,\ldots,n\}$ se calcula $x'_i(t+1)$ con la matriz de recuperación pasando posteriormente por la función de activación de acuerdo con el siguiente criterio:

$$x'_i(t+1) = \begin{cases} 1 & si \sum_{j=1}^{n} w_{ij}\, x'_j(t) > \theta \\ x'_i(t) & si \sum_{j=1}^{n} w_{ij}\, x'_j(t) = \theta \\ -1 & si \sum_{j=1}^{n} w_{ij}\, x'_j(t) < \theta \end{cases}$$

Ecuación 2-6: Cálculo 1 de recuperación

3. Se compara $x'_i(t+1)$ con $x'_i(t) \forall i \in \{1,2,..,n\}$. Si $x'(t+1) \approx x(t)$ el proceso termina. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar a un tiempo t=τ para el cual $x(\tau+1) \approx x(\tau)$; el proceso termina y el patrón recuperado es $x(\tau)$.

El proceso de convergencia descrito en el paso 3 de la fase de recuperación, indica que el sistema llega un punto límite local establecido previamente con el tiempo τ. Este constante garantiza a través de la demostración que hace Hopfield de la existencia de un punto límite local estable en su modelo de memoria asociativa. Para ello, define una función de la siguiente manera, tomando estas condiciones: $w_{ij} = 0$ si i=j.

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\, x_i x_j$$

Ecuación 2-7: Función de energía

Donde demuestra que E es una función de x_i monótona decreciente.

2.4 Función de activación

En redes computacionales, la función de activación de un patrón define la salida de un patrón de entrenamiento o de un conjunto de patrones. Se suelen distinguir entre funciones lineales, en las que la salida es proporcional a la entrada; funciones de umbral, en las cuales la salida es un valor discreto que depende de si la estimulación total supera o no un determinado valor de umbral; y funciones no lineales, no proporcionales a la entrada.

Ejemplo de algunas funciones:

- Función sigmoidal (lineal): probablemente es la función de activación más utilizada en la actualizada. Se trata de una función continua no lineal comprendido en un rango de mínimo 0 y máximo 1.

$$\text{f. sigmoidal(z)} = \frac{1}{1 + e^{-z}}$$

Ecuación 2-8: Función sigmoidal

- Función gausiana (lineal): la primitiva de una función gaussiana es la función error. Esta utiliza constantes reales para su cálculo.
    - Siendo a, b, c constantes reales (c > 0)

$$f(z) = a\, e^{-\frac{(z-b)^2}{2c^2}}$$

Ecuación 2-9: Función gaussiana

- Función umbral (no lineal): más conocida en inglés como threshold function, es una función booleana monótona.

$$f(x) = \begin{matrix} 1\ si \sum wx \geq umbral \\ 0\ si \sum wx < umbral \end{matrix}$$

Ecuación 2-10: Función umbral no lineal

- Función logística (lineal): Es la versión continua de la función umbral y se utiliza en problemas de aproximación. Es continua a valores en [0,1] e infinitamente diferenciable.

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

Ecuación 2-11: Función logística

- Función de signo (no lineal): es una función matemática definida a trozos, que obtiene el signo de cualquier número real que se tome por entrada. Se representa generalmente mediante sgn(x).

$$sgn(x) = \begin{matrix} 1\ si\ x > 0 \\ 0\ si\ x = 0 \\ -1\ si\ x < 0 \end{matrix}$$

Ecuación 2-12: Función de signo

3 Diseño y desarrollo

3.1 Diseño y arquitectura del sistema

En este capítulo se diseñará de manera más técnica la biometría de las retinas después de las bases teóricas explicadas en el capítulo anterior comenzaremos nuestra investigación práctica a partir de la memoria asociativa Hopfield. Los patrones fundamentales que utilizaremos como neuronas de entradas serán imágenes pixeles a blanco y negro de tamaño n*n cuadráticas. Las implementaciones posteriores se realizarán en un lenguaje de programación de alto nivel: **C**; y en Matlab.

3.1.1 Módulos de la funcionalidad

A continuación, se mostrará una lista de los requisitos de la funcionalidad que deberá tener la aplicación. Se dividirá el trabajo en pequeños módulos para facilitar el trabajo de implementación posterior.

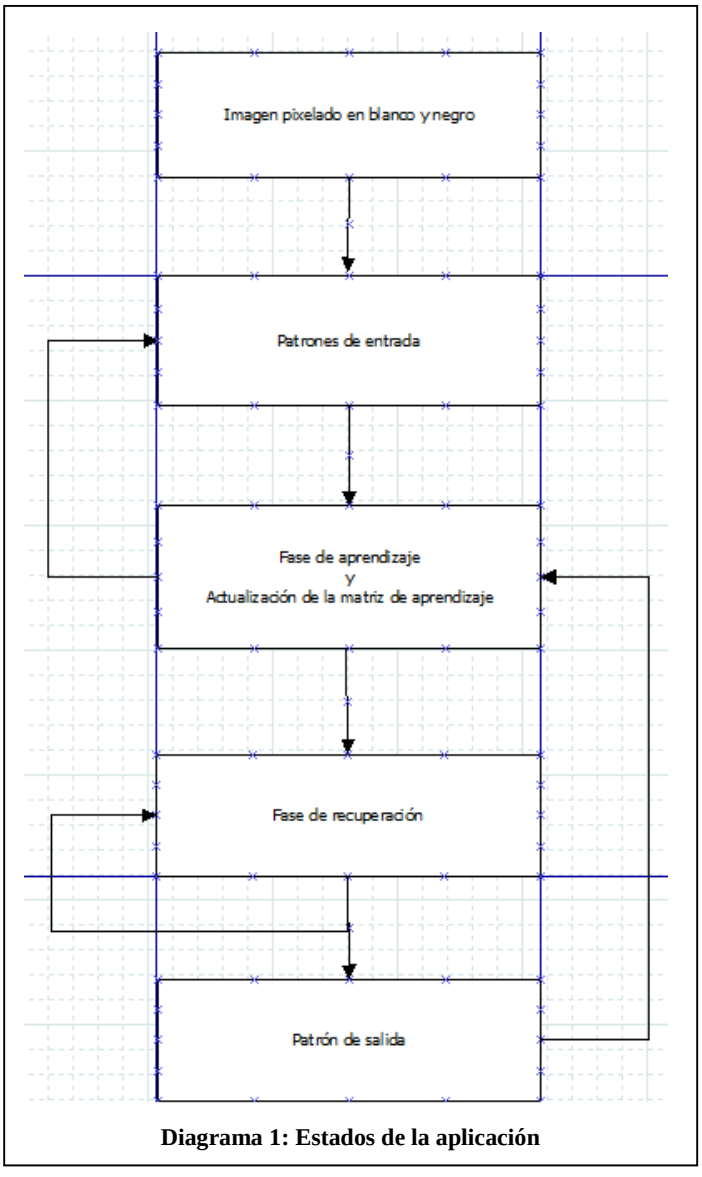


Diagrama 1: Estados de la aplicación

- **Procesamiento de imágenes.** Como se ha mencionado anteriormente, los patrones que utilizamos en nuestra investigación son imágenes cuadradas en blanco y negro. Utilizaremos una herramienta externa para transformar dichas imágenes en valores binarios o de signo.
- **Importación de patrones de entrada.** De nuestro conjunto de imágenes procesadas anteriormente, agruparemos por aquellas que tengan más relevancia para nuestro estudio, ya que un aprendizaje no interrumpido puede afectar en consecuencias nuestros resultados generados.
- **Fase de aprendizaje.** Mediante los algoritmos de la memoria asociativa de Hopfield, generamos la matriz de pesos calculados a partir de los patrones de entradas selectivas.
- **Actualización de la base de conocimiento.** Esa matriz obtenida anteriormente se adaptará de forma automática a los nuevos patrones introducidos, calculando de forma que la matriz individual sea acumulada a una base de conocimiento aprendido anteriormente.

- **Fase de recuperación.** A partir de nuestra base de conocimiento introduciremos un patrón con ruido para la prueba de nuestra investigación. Se puede decir a priori que nunca tendríamos la garantía de saber si la recuperación ha sido perfecta al 100%, es decir, si no sabíamos cómo era el patrón originalmente ya que el nuevo patrón recuperado es una aproximación a ella.
    - Cabe destacar uno de los puntos más relevantes de la investigación en este módulo, el proceso de la función de activación. Este proceso regulará los resultados obtenidos en la fase de recuperación un patrón de salida con un valor similar a los patrones fundamentales.
- **Patrón de salida.** Los resultados obtenidos nos facilitan entender el funcionamiento de nuestro sistema. Estos pueden ser tratados nuevamente como un nuevo patrón fundamental para las pruebas posteriores.
- **Procesamiento inverso a imágenes.** Los resultados obtenidos en la fase de recuperación son valores binarios o de signo donde posteriormente se utilizará la herramienta de Matlab para transformarlos a imagen.

3.1.2 Estructura de la arquitectura

En esta sección definiremos la estructura de almacenamiento en memoria dinámica de los patrones y pesos generados:

- **Patrón**: patrón de entrada o salida.
    - →pixel: puntero entero de valores binarios o de signo de los patrones de entrada o salida.
    - →SumPixeles: entero del sumatorio de los valores del patrón de entrada o salida. Control simple de si hubo modificación o no respecto del estado anterior.
- **Peso**→matriz: puntero doble entero del matriz peso generado en la fase de aprendizaje. Esta matriz es actualizada por cada patrón fundamental obtenida.
- **Elemento**: unión de patrón con su matriz peso individual.
    - →dato: patrón
    - →pesow: matriz peso individual.
    - →siguiente: conexión con otros patrones.
- **Lista**: Conexión total de todos los patrones fundamentales. Conjunto.
    - →inicio: primer elemento del conjunto.
    - →final: último elemento del conjunto.
    - →pesoConjunto: matriz total de fase de aprendizaje generado por todos los patrones fundamentales.
    - →cantidad: número de patrones de entrada.

3.2 Desarrollo de la arquitectura del sistema

El formato de las imágenes x^{μ} que comenzaremos a procesar tiene una dimensión n*n, trataremos cada 1´s (pixeles blancos) de esos n pixeles de la imagen como una neurona del patrón de entrada. La dimensión de los patrones fundamentales que establecemos puede afectar al resultado visual de nuestro resultado, pero esto también va a depender mucho del rendimiento de nuestra computadora y del sistema operativo. Por lo que reduciremos el tamaño de estas imágenes a un tamaño más pequeño en base a 2 para facilitarnos el cálculo. El tamaño más aceptable que podría tomar puede ser de 256x256, pero esto tendría un total

de 65.536 componentes de entrada, aproximadamente 64 Megas de memoria por cada patrón de entrada. Con una matriz de tamaño $(256x256)^2$= 4.294.967.296 ≈ 4 Gigas de memoria dinámica. Este resultado puede suponernos a la hora de trabajo muy costoso. Para nuestra investigación hemos decidido utilizar un factor de 128x128, 16.384 elementos de entrada (16 Megas). Los pesos que obtendremos tendrán un tamaño de 256 Megas.

La reducción de una imagen supone pérdidas de información. La mayor parte de nuestras imágenes tienen aproximadamente un 20 % de pixeles blanco respecto a un 80 % de pixeles negros, para no quedarnos *"a oscuras"* dilataremos[6] la imagen original que posteriormente será reducida.

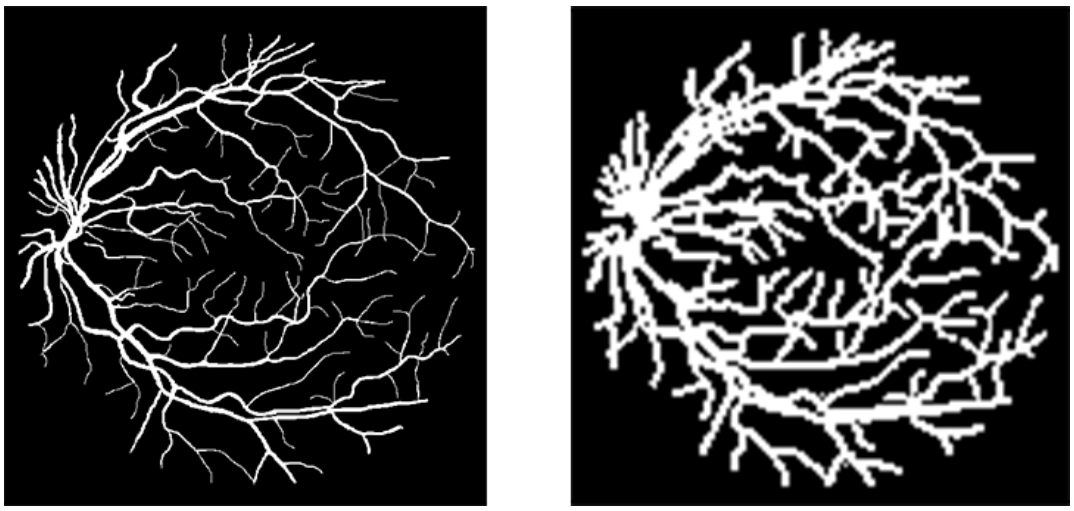


Imagen 2: Retina original (izquierda) y retina modificada 128x128 (derecha)

En base a las ecuaciones estudiados en la sección de *Estado del arte* hemos realizado algunas modificaciones del algoritmo para mejorar la eficiencia de tiempo y rendimiento en las computadoras. Los dos algoritmos de la memoria de Hopfield que haremos la mejora serán: para la fase de aprendizaje, la matriz de peso (***Ecuación 1***); y para la fase de recuperación, la ecuación de recuperación (***Ecuación 5***).

La matriz peso de la teoría hebbiana w_{ij} es una matriz obtenida a partir de la multiplicación del vector del patrón de entrada por su vector transpuesto, el cálculo computacional de esta matriz tiene un coste de n^2 siendo *n* el número de elementos del patrón, donde en nuestro caso es igual a 16.384, con un coste total de más de 268 millones de operaciones[7]. Sabiendo con antelación que el valor de w_{ij} es una multiplicación de los elementos de la posición i y j, hacemos uso de la propiedad conmutativa de la multiplicación *"el orden de los factores no altera el producto"* sabemos que dos valores de la matriz siempre son iguales $w_{ij} = w_{ji}$. Otra propiedad que cumple también la matriz del peso de la teoría hebbiana es que el valor de la conexión de un elemento consigo mismo es siempre nulo. Teniendo en cuenta las dos propiedades anteriores, el nuevo seudocódigo que utilizaremos para el sistema ha sido la

[6] **Dilatación**: (Dilatar) Extender, alargar y hacer mayor algo, o que ocupe más lugar o tiempo.

[7] El número de operaciones es calculado a partir del algoritmo computacional de los sistemas informáticos utilizando el conocimiento de análisis de algoritmo.

siguiente, con un nuevo coste computacional donde será $\frac{(n*(n+1))}{2} - \#op.matrizID = 134.225.920 - 16.384$ aproximadamente 134 millones de operaciones, reduciendo el coste a la mitad.

```
para i desde 1 hasta 128
    para j desde i+1 hasta 128
        MatrizPeso (i,j) = Xi * Xj
```

Tras haber modificado el algoritmo de la fase de aprendizaje, la matriz resultante ya no puede ser utilizada en la fase de recuperación porque en algunas posiciones este valor es igual a cero o nulo. Existirían dos posibilidades de reparar este hecho:

1. Completar en las posiciones nulas con valores de las posiciones transpuestas, donde de esta forma no sería necesario modificar el algoritmo de la fase de recuperación.
2. Cambiando el algoritmo de la fase de recuperación haciendo uso de las posiciones con valores buenos.

```
para i desde 1 hasta 128^2
    para j desde 1 hasta 128^2
        si i = j entonces
            VSPrima(i) = 0
        si j > i entonces
            VSPrima(i) = MatrixPeso(i,j) * VEntrada(j)
        si no
            VSPrima(i) = MatrixPeso(j,i) * VEntrada(j)
```

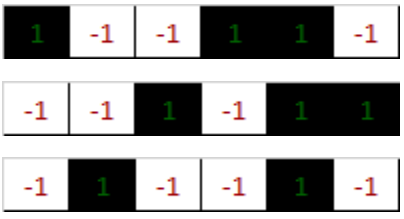
Desde el punto de vista del análisis de algoritmo de la programación la opción 1 es el mejor porque el tiempo de procesamiento de una asignación de variables es más rápida que el tiempo que tarda en la comparación de dos variables (es este caso denominamos el factor tiempo como el número de operaciones lógicas).

Comenzamos aquí el análisis de la memoria asociativa de Hopfield, pero para simplificar la documentación de este trabajo, ya que no es factible analizar de forma manual un patrón de entrada de tamaño 128x128, utilizaremos patrones con una dimensión n=6 simularemos los pasos que seguirían nuestro sistema para el cálculo de la fase de aprendizaje y en la fase de recuperación. Los patrones fundamentales que utilizaremos de prueba serán los siguientes:

$$x^1 = (1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1)$$

$$x^2 = (-1 \quad -1 \quad 1 \quad -1 \quad 1 \quad 1)$$

$$x^3 = (-1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1)$$



Calcularemos, mediante las expresiones ***Ecuación 2*** y ***Ecuación 3*** obtenemos las siguientes matrices individuales:

$$wx^1 = x^1(x^1)^t - \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & -1 & 1 & 1 & -1 \\ -1 & 0 & 1 & -1 & -1 & 1 \\ -1 & -1 & 0 & -1 & -1 & 1 \\ 1 & -1 & -1 & 0 & 1 & -1 \\ 1 & -1 & -1 & 1 & 0 & -1 \\ -1 & 1 & 1 & -1 & -1 & 0 \end{pmatrix}$$

$$wx^2 = x^2(x^2)^t - \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 & 1 & -1 & -1 \\ 1 & 0 & -1 & 1 & -1 & -1 \\ -1 & -1 & 0 & -1 & 1 & 1 \\ 1 & 1 & -1 & 0 & -1 & -1 \\ -1 & -1 & 1 & -1 & 0 & 1 \\ -1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix}$$

$$wx^3 = x^3(x^3)^t - \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & 1 & -1 & 1 \\ -1 & 0 & -1 & -1 & 1 & -1 \\ 1 & -1 & 0 & 1 & -1 & 1 \\ 1 & -1 & 1 & 0 & -1 & 1 \\ -1 & 1 & -1 & -1 & 0 & -1 \\ 1 & -1 & 1 & 1 & -1 & 0 \end{pmatrix}$$

El peso de la memoria de Hopfield se obtiene a partir de la suma de las matrices obtenidas anteriormente que corresponde al tercer paso de la fase de aprendizaje:

$$w = wx^1 + wx^2 + wx^3 = \begin{pmatrix} 0 & -1 & -1 & 3 & -1 & -1 \\ -1 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & 3 \\ 3 & -1 & -1 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & 3 & -1 & -1 & 0 \end{pmatrix}$$

Con la obtención de la memoria w se concluye la fase de aprendizaje. Ahora utilizaremos el patrón $x' = x^3$ para llevar a cabo la fase de recuperación. El primer paso consiste en realizar la asignación $x^3 = x'(0)$. Entonces se tiene:

$$x'(0) = x^3 = (-1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1)$$

-1	1	-1	-1	1	-1

Aplicando la condición del paso 2 de la fase de recuperación y posteriormente la función de activación, el valor del umbral que aplicaremos en este caso será el número cero, obtenemos el siguiente resultado:

$$x'(1) = w * x'(0) = (-3 \quad 3 \quad -3 \quad -3 \quad 3 \quad -3)$$

$$x'(1) = w * x'(0) = (-1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1)$$

-1	1	-1	-1	1	-1

Con el paso 3 se realizan las comparaciones de $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1,2..n\}$:

$$x'_1(1) = -1 = x'_1(0)$$
$$x'_2(1) = 1 = x'_2(0)$$
$$x'_3(1) = -1 = x'_3(0)$$
$$x'_4(1) = -1 = x'_4(0)$$
$$x'_5(1) = 1 = x'_5(0)$$
$$x'_6(1) = -1 = x'_6(0)$$

Podemos observar que el resultado obtenido en la $x'(1)$ es igual al patrón fundamental de entrada $x'(0)$ por lo que damos por hecho el éxito de la recuperación de un patrón de entrada que previamente se ha aprendido de ella. El éxito de esta recuperación no significa que en

todos los casos darán los mismos resultados, utilizando el patrón fundamental 1 como patrón de recuperación e introduciéndole ruido realizaremos la siguiente prueba, como la matriz de aprendizaje ya está calculada no volvemos mostrarla.

$$x'(0) = (1 \quad -1 \quad -1 \quad -1 \quad 1 \quad -1)$$

Aplicando la condición del paso 2 de la fase de recuperación y la función de activación umbral se obtiene el siguiente resultado:

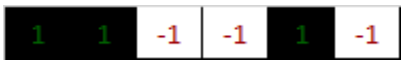
$$x'(1) = w * x'(0) = (-1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1)$$

El paso 3 de la fase de recuperación indica que se realicen las comparaciones de $x'(t+1)$ con $x'(t)$:

$$x'_1(1) = -1 \neq x'_1(0)$$
$$x'_2(1) = 1 \neq x'_2(0)$$
$$x'_3(1) = -1 = x'_3(0)$$
$$x'_4(1) = 1 = x'_4(0)$$
$$x'_5(1) = 1 \neq x'_5(0)$$
$$x'_6(1) = -1 = x'_6(0)$$

Podemos observar que los resultados obtenidos en el patrón de salida $x'(1)$ para las posiciones i = 1, 2 y 5 no han coincidido con los valores originales del patrón, eso quiere decir que la recuperación del patrón de entrada no está en un estado estable por lo que se repetirá los pasos 2 y 3 hasta que llegue a un estado más estable o repetitivo. Volviendo a aplicar el paso 2 de la fase de recuperación y la función de activación tenemos para el patrón $x'(2)$ en la segunda iteración el siguiente resultado:

$$x'(2) = w * x'(1) = (1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1)$$



El resultado obtenido sigue sin coincidir con el patrón fundamental x_1 lo que hace necesario que se vuelva a repetir la fase de la recuperación en una 3ª iteración:

$$x'(3) = w * x'(1) = (-1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1)$$

El paso 3 de comparación de valores en los estados de $x'(3)$ con $x'(2)$ tampoco ha dado un resultado positivo, pero si recordásemos el resultado obtenido en la primera iteración $x'(1)$, este patrón es idéntica a ella por lo que la fase de repetición había entrado en un estado de bucle infinito. Todas estas teorías matemáticas de recuperación se han obtenido con trampa porque estamos observando los resultados *"a ojo"*, en una ejecución real del programa desarrollado nosotros no tendríamos por qué saber cómo es el patrón original, ni tampoco sabríamos si la fase de recuperación ha entrado en una repetición.

PREGUNTA: *¿Cuántos estados diferentes tendríamos que almacenar para calcular su nivel de estabilidad?*

3.3 Punto de activación (umbral)

La función de activación que hemos estado utilizando en nuestro desarrollo de la memoria asociativa de Hopfield ha sido la siguiente:

$$x_i(t+1) = \begin{Bmatrix} 1 & si \sum_{j=1}^{n} m_{ij}\, x_j(t) > \text{umbral} \\ x_i(t) & si \sum_{j=1}^{n} m_{ij}\, x_j(t) = \text{umbral} \\ -1 & si \sum_{j=1}^{n} m_{ij}\, x_j(t) < \text{umbral} \end{Bmatrix}$$

Ecuación 3-1: Función de activación

El umbral utilizado para su cálculo ha sido el valor cero, donde los posibles valores de la memoria de Hopfield pertenece a $\{-1, +1\}$. Este umbral en un principio era correcto ya que pertenece al valor intermedio de los dos posibles estados, pero después de realizar nuestro estudio hemos visto que el umbral utilizado no es el mejor ya que más del 70% de nuestra imagen pertenece a un valor negativo, es decir, tiene el color negro. Para resolver el problema del umbral hemos realizado otros tres tipos umbrales mediante diferentes algoritmos.

3.3.1 Cálculo del umbral 1: media aritmética

El primer tipo de umbral adicional que hemos utilizado se basa en el uso de la media aritmética de la matriz de peso w y patrón de recuperación $x´$. Este umbral tiene en cuenta el tamaño de la dimensión N del patrón de recuperación, el número P de patrones fundamentales aprendidos anteriormente y B^3 siendo el porcentaje de la diferencia de número de elementos blancos y número de elementos negro entre el número total de elementos del patrón. Es decir, el cálculo de este umbral utiliza las siguientes ecuaciones:

$$Umbral\ \theta = N * P * b^3$$

Ecuación 3-2: Cálculo 1 umbral 1

$$b = \frac{(\#blancos\ -\#negros)}{N}$$

Ecuación 3-3: Cálculo 2 umbral 1

Haciendo ejemplo de nuestra investigación, la mayoría de los patrones fundamentales tienen aproximadamente 30% de elementos blancos de un total de 128^2 elementos, si realizamos un aprendizaje de 20 patrones fundamentales entonces nuestro umbral tiene un valor aproximado de $Umbral\ \theta = 20.971{,}52$

```
function Umbral = BuscarUmbral3(PatronE,NumPatrones,Longitud)
    LongCuadrado = Longitud * Longitud;
    Blanco = sum(sum((PatronE==1)*1));
    Negro = sum(sum((PatronE==-1)*1));
    B = (Blanco-Negro)/LongCuadrado;
    B = abs(B*B*B);
    Umbral = LongCuadrado * NumPatrones * B;
```

Imagen 3: Seudocódigo MatLab 1

3.3.2 Cálculo del umbral 2: bisección

El algoritmo utilizado para calcular el umbral 2 ha sido de *prueba y error*, es decir, una vez obtenida el patrón de recuperación después de la fase de recuperación se procesará por la función de activación mediante un umbral calculado de forma *personalizada*. El valor del umbral $\theta \in [-N^2, +N^2]$ donde si N=128 entonces los posibles valores del umbral pertenece a $[-16384, +16384]$, es decir, si todos los elementos de un patrón perteneciera al color blanco entonces la suma de valores del patrón es igual al máximo de la suma de 1´s del patrón.

La técnica que hemos utilizado para la búsqueda del umbral ha sido el método de la bisección, limitaremos los posibles valores del umbral en un máximo y mínimo, dividiremos el rango en dos secciones, el umbral de la sección 1 del $[mínimo, media]$ y el umbral de la sección 2 del $[media, máximo]$, calcularemos el mejor resultados de los dos umbrales y se irá ajustando tendiendo a la sección con el mejor resultado.

```
function Umbral = BuscarUmbral2(PatronO,PatronR,Longitud)
    maxUmbral = Longitud * Longitud;
    minUmbral = 0;
    mitadUmbral = (minUmbral + maxUmbral)/2;
    Error = 100.0;
    while ((Error > 10) && ((maxUmbral-minUmbral) > 0.1))
        mitad1 = (minUmbral + mitadUmbral)/2;
        mitad2 = (mitadUmbral + maxUmbral)/2;

        Temp1 = FuncionActivacion1(PatronR,mitad1);
        Error1 = PorcentajeError(PatronO,Temp1,Longitud*Longitud);
        Temp2 = FuncionActivacion1(PatronR,mitad2);
        Error2 = PorcentajeError(PatronO,Temp2,Longitud*Longitud);
        if Error1 <= Error2
            maxUmbral = mitadUmbral;
            UmbralAux = mitad1;
            Error = Error1;
        else
            minUmbral = mitadUmbral;
            UmbralAux = mitad2;
            Error = Error2;
        end;
        mitadUmbral = (minUmbral + maxUmbral)/2;
    end
    Umbral = UmbralAux;
```

Imagen 4: Seudocódigo MatLab 2

El porcentaje máximo de error permitido que hemos establecido ha sido de un 10%, nuestro algoritmo irá ajustando el valor del umbral hasta obtener resultado máximo del porcentaje establecido.

3.3.3 Cálculo del umbral 3: por región

Otro de los algoritmos que hemos utilizado para solucionar el problema de la divergencia de los valores del patrón de recuperación hacia el lado negativo es hacer uso de la frase *"Divide y vencerás"*, separando la imagen en pequeñas regiones con un tamaño múltiple a la dimensión de la imagen y calculando el número de elementos blancos que existe en ella. De esta manera no tendríamos un solo umbral, sino tantas como regiones hemos separado, ya que no es igual un umbral de las esquinas de la imagen, región totalmente negativo, que el umbral de una zona de imagen con valor blanco.

La dimensión utilizada para las nuevas regiones del patrón de recuperación tiene un tamaño de n=8 formando así en un total de 16x16=256 regiones individuales, es decir, los elementos de las posiciones del $i \ni \{1,2,3,4,5,6,7,8\}$ y $j \ni \{1,2,3,4,5,6,7,8\}$ pertenecerá a la nueva matriz individual mt_1 y así sucesivamente. Con esta modificación de la función de

activación cada región utilizará su propio umbral para calcular el valor estimado, de esta forma, evitamos que los resultados recuperados tengan una divergencia al valor negativo.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16
Y1	0	0	0	0	0	0	0	2	4	2	0	0	0	0	0	0
Y2	0	0	0	0	3	19	0	17	49	33	13	0	0	0	0	0
Y3	0	0	0	20	24	25	25	45	45	41	32	38	25	1	0	0
Y4	0	5	11	18	52	54	34	22	32	16	26	28	12	1	0	0
Y5	0	25	22	47	35	13	8	27	14	31	23	19	14	20	14	0
Y6	3	49	51	29	23	23	22	16	29	25	22	25	36	25	7	0
Y7	3	41	58	38	31	38	36	26	35	35	30	34	24	18	14	5
Y8	20	49	41	32	29	38	27	7	15	23	21	31	27	25	17	17
Y9	20	37	29	11	27	11	21	17	8	28	27	18	16	5	6	17
Y10	2	26	33	43	33	17	31	25	20	36	29	14	21	18	10	2
Y11	4	30	33	38	35	22	22	37	26	26	18	21	29	21	28	4
Y12	0	12	16	10	44	40	23	16	23	23	22	19	21	35	11	0
Y13	0	0	3	12	26	46	37	29	27	22	35	38	24	29	2	0
Y14	0	0	0	0	24	26	5	13	37	32	14	29	21	4	0	0
Y15	0	0	0	0	7	16	27	19	26	31	10	1	6	0	0	0
Y16	0	0	0	0	0	0	6	0	9	7	0	0	0	0	0	0

Tabla 2: Tabla de regiones

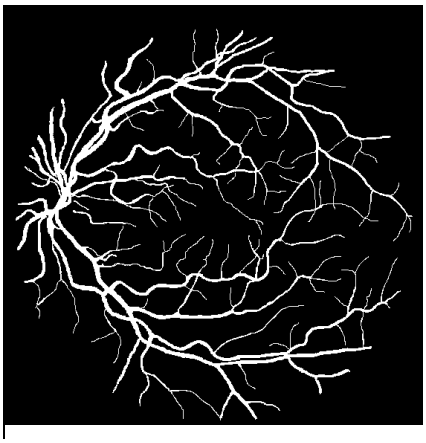
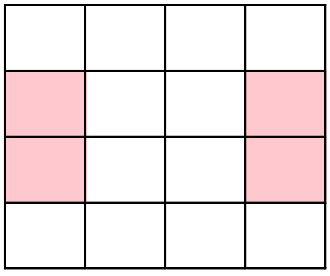


Imagen 5: Retina de entrenamiento

La tabla de regiones calculada anteriormente ha sido obtenida a partir de la retina de entrenamiento de la derecha, podemos observar que todas retinas tienen una orientación lateral, en nuestro caso, esta imagen está orientada hacia la izquierda, esto puede ser importante a la hora de establecer los umbrales de la función ya que no es igual poner un valor de un umbral de una zona blanca que un umbral de una zona negra.

Para el cálculo de la tabla se realizará los siguientes pasos:

1. Determinar la orientación de la imagen: este paso se calcula a partir de la cantidad acumulada de elementos blancos concentrados en una región determinada poniendo como referencia el lado derecho e izquierdo.



Las zonas resultadas de la imagen del cuadrado serán las zonas que exploraremos en busca de la orientación de la imagen, si el lado izquierdo tiene más elementos blancos, entonces la imagen esta orientada hacia la izquierda, en caso contrario, esta orientada hacia la derecha.

2. Calcular por cada patrón fundamental el umbral de cada región: se repartirá la imagen en tantas regiones como se desee repartir, pero siempre es recomendable que sea un

valor múltiple en base a 2 y divisor de n, siendo n el tamaño del patrón fundamental. Si la imagen es repartida en n=4, entonces tendríamos un total de 16 regiones, siendo $TamSec = 128/16$

a. Sección de la izquierda:

$$SubSeccion((TamSec + 1) : (TamSec * 3), 1 : TamSec);$$

b. Sección de la derecha:

$$SubSeccion((TamSec + 1) : (TamSec * 3), (TamSec * 3 + 1) : (TamSec * 4));$$

3. Acumular en una matriz el umbral u_{ij} el promedio del umbral de todos los patrones fundamentales.
4. Después de la fase de recuperación de la memoria de Hopfield se procesará por la función de activación eligiendo el umbral de la región perteneciente:

$$umbral_i = (\frac{(i-1)}{n} + 1)$$

$$umbral_j = (\frac{(j-1)}{n} + 1)$$

$$x_{ij} => umbral = U(umbral_i, umbral_j)$$

En la tabla siguiente hemos dividido la imagen en n=16 en un total de 256 regiones diferentes, los valores que se muestran en la tabla pertenece al número de elementos blancos existentes en cada región individual. El color rojo muestra la zona que apenas aparecen pixeles blancos o con algunos de ellos que posiblemente pueden ser errores. Y la zona azul son las regiones que muestran mayor intensidad de elementos blancos. Mediante esta tabla podemos observar fácilmente, de forma numérica, la distribución de los elementos de un patrón de entrada o salida.

3.3.4 Comparación de diferentes tipos de umbral

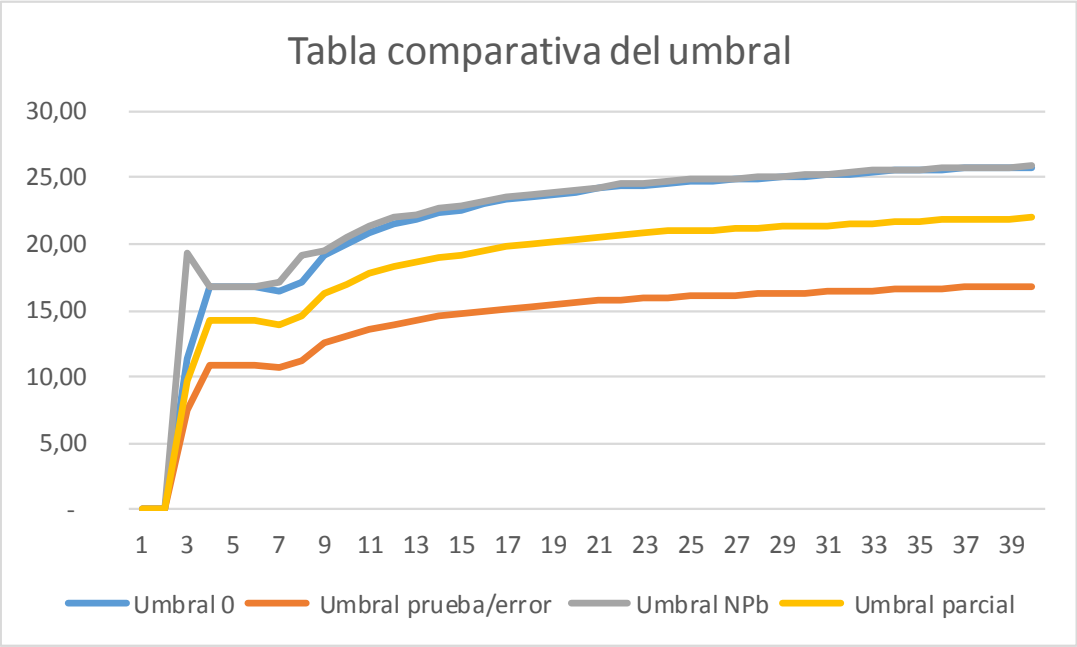


Tabla 3: Gráfica comparativa del umbral

Después de la investigación e implementación de los diferentes tipos de umbrales, comenzaremos comentado las ventajas e inconveniencias de cada una de ellas:

- Función umbral cero: el valor 0, que se ha utilizado como umbral predeterminado de la función de activación, no es un valor aleatorio que se ha obtenido desde la nada, sino que es el resultado de la media aritmética del valor máximo y mínimo de los posibles valores de $x_i \in (-1,1)$. Este tipo de umbral es bueno si el número de elementos blancos y el número de elementos negros es proporcional. En nuestra investigación de la biometría no nos ha resultado útil ya que la proporción de elementos blancos y negros han sido de un 30% y 70% respectivamente, por lo que la imagen tiende al lado negativo.

- Función umbral bisección (prueba y error): el valor del umbral de este tipo de función de activación tiene un valor numérico diferente en cada patrón de recuperación. No existe ningún algoritmo que dé el resultado del mejor umbral, sino que realizamos el ajuste *manual* mediante aproximación de un rango máximo y mínimo, actualizando su valor en cada iteración. Pero este tipo de umbral se calcula con *trampa* porque hemos dado por supuesto la existencia de un patrón fundamental original bueno con la que realizamos comparación con el patrón recuperado hasta conseguir un resultado óptimo.

- Función umbral NPb3: este umbral, similar al umbral cero, es calculado mediante la media aritmética, los factores utilizados han sido: el número de patrones fundamentales aprendidos, el número de elementos existentes en cada patrón de entrada y el porcentaje de elementos blancos y negro del patrón de recuperación. En nuestra investigación este valor no ha resultado totalmente bueno ya que los pixeles blancos están concentrados en la zona interior de la imagen. El umbral conoce el posible valor que ha de tomar, pero no siempre es correcto ya que depende del lugar en que está situado el elemento.

- Función umbral parcial: el peso del umbral de esta función de activación se obtiene repartiendo la imagen en diferentes secciones, por lo que conoce mejor la concentración de los pixeles blancos y negros. Pero existe una desventaja sobre este tipo de función, el umbral parcial es calculado sobre un patrón de recuperación, dicho patrón puede estar *contaminada* por diferentes valores, si una región es eliminada o completada totalmente, su umbral de recuperación puede ser 0% de probabilidad de ser blanco o 100% de probabilidad de blanco.

Mediante los resultados obtenidos en la ***Tabla 3: Gráfica comparativa del umbral***, la mejor función de activación ha resultado ser la funcion de biyección. Este resultado se ha mantenido hasta el aprendizaje de más de 10 patrones fundamentales, parece ser, como comentábamos en los párrafos anteriores, que la matriz de peso y el patrón de recuperación ha sido *contaminada*. Excluyendo la función de biyección, la siguente función de activación con el mejor umbral ha sido la función de umbral parcial, siempre que consideremos que el patrón a recuperar no está inclustada con un ruido parcialmente total.

4 Integración, pruebas y resultados

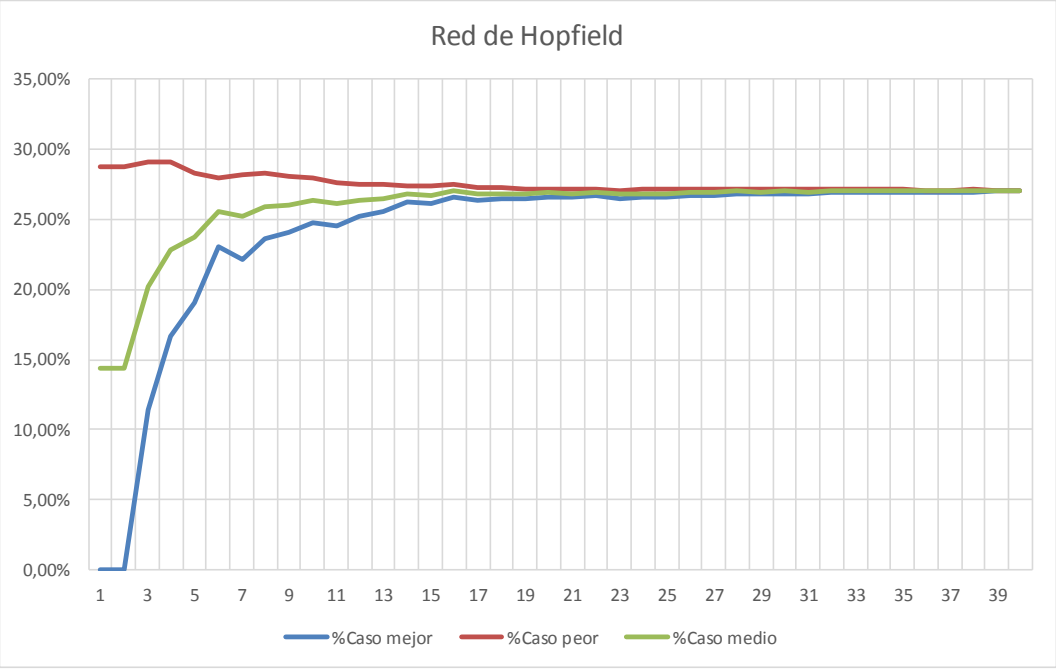
4.1 Resultados

El número de patrones utilizadas en esta investigación han sido de un máximo de 40 patrones de entrada de aprendizaje y de hasta 20 patrones de recuperación. Se han obtenido todo tipo de resultados en la investigación, tanto buenos como malos, a continuación describiremos el motivo del éxito y del fracaso.

4.1.1 Red de Hopfield

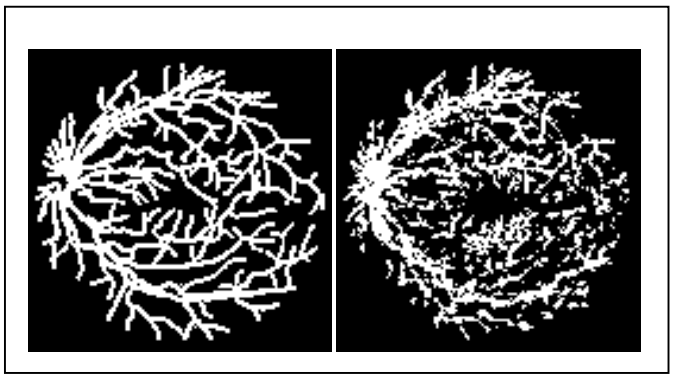
La red de Hopfield es una red de adaptación probabilística que pertenece a la memoria auto asociativa, es decir, memorizan nuevos patrones durante el momento de entrenamiento. Haciendo hincapié en esta característica, hemos realizado tres tipos de prueba dependiendo de los patrones de entrada para la fase de aprendizaje:

- Caso mejor: todos los patrones de la retina poseen una orientación lateral, es decir, con una orientación izquierda o derecha, teniendo un conocimiento previo a ella, hemos escogido patrones únicamente con las mismas orientaciones. Los resultados obtenidos han sido óptimos en los cinco patrones de entrada, donde el porcentaje de error de los elementos era inferior al 20%, a partir de los siguientes patrones el porcentaje de errores tiende a un límite superior del 27,5% aproximadamente.
- Caso peor: a diferencia del caso mejor, en esta prueba hemos optado por escoger patrones de entrada con una orientación contraria, los primeros resultados obtenidos podrían decirse que son pésimos, haciendo ejemplo de un patrón de retina de tamaño 256x256 elementos, más de 19000 elementos de ella han entrado en conflicto por los resultados del patrón de salido, pero después de un tiempo de aprendizaje, el número de errores comienza a estabilizarse con el mismo porcentaje que el caso mejor.
- Caso medio: para obtener unos resultados más convincentes, en este caso, hemos escogido patrones al azar para la fase de aprendizaje.



Después de analizar los patrones de entrada y visualizando los datos obtenidos de la tabla anterior, hemos obtenido un resultado de aproximadamente 27,50% en el porcentaje de errores de la red de Hopfield para el aprendizaje y recuperación de la biometría de las retinas. Este valor numérico que hemos obtenido no lo consideramos como un buen algoritmo para nuestro estudio, ya que el porcentaje de elementos de valor negativo (zonas con color negro) abarca más del 75% de la imagen de un patrón de la retina, eso hace que los valores de los elementos tienden más al color oscuro por la interconexión total de los elementos del patrón.

4.1.1.1 Experimento 1



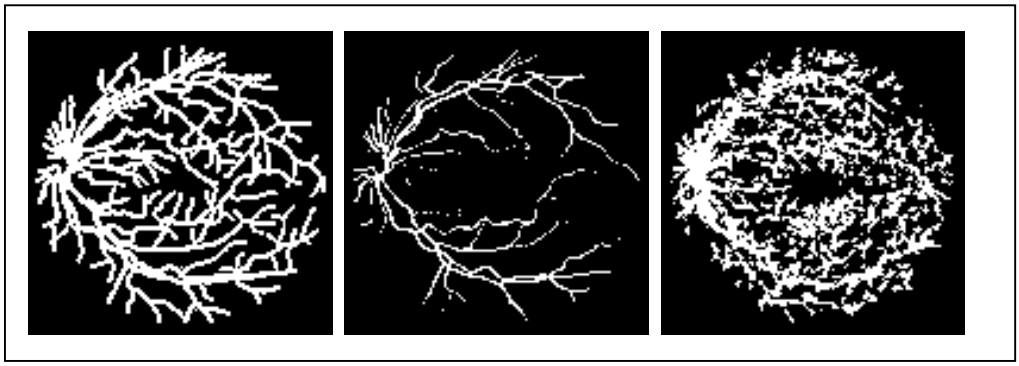
Experimento 1: recuperación de una imagen aprendida en la fase de aprendizaje.

Imagen 1, patrón original; Imagen 2, patrón de salida.

Diferencia 1-2: 13.34 %.

Podemos observar que en este caso la diferencia que hubo entre imagen uno y la imagen dos es de más del 10%. Aun intentando recuperar la imagen aprendido el factor de error no será cero si previamente ha aprendido una base de conocimiento aparte de otros patrones de entrada.

4.1.1.2 Experimento 2



Experimento 2: recuperación de una imagen con ruido de una imagen aprendida en la fase de aprendizaje.

Imagen 1, patrón aprendido; imagen 2, patrón a recuperar; imagen 3, patrón recuperado.

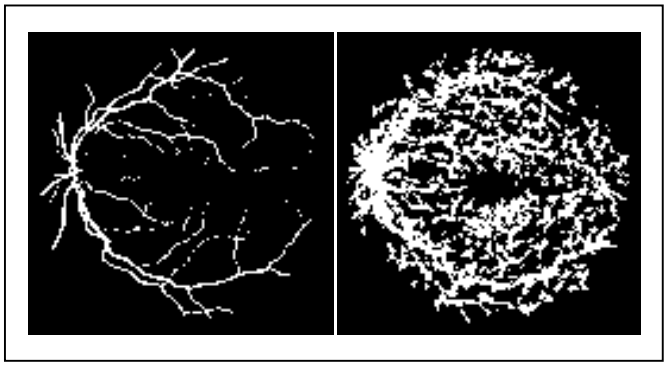
Diferencia 1-2: 19.28 %.

Diferencia 2-3: 23.39 %.

Diferencia 1-3: 21.15 %.

En este caso le hemos escogido una imagen como una diferencia de error de aproximadamente un 20 %. A partir de nuestra fase de aprendizaje hemos reducido en un 2 % la diferencia que hubo de la imagen original y la imagen recuperada.

4.1.1.3 Experimento 3



Experimento 3: recuperación de una imagen desconocida por la memoria asociativa. Imagen 1, patrón a recuperar; imagen 2, patrón recuperado.

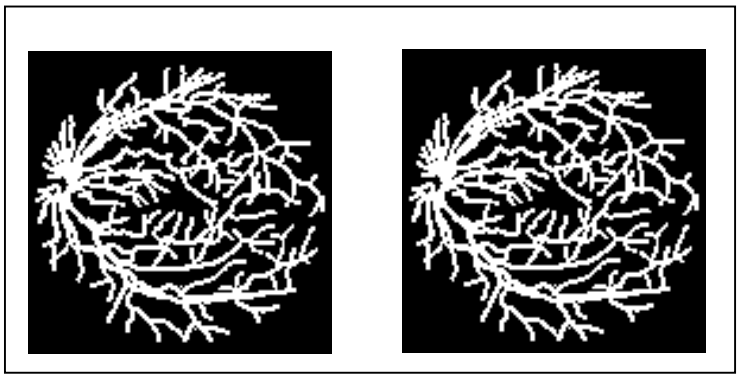
Diferencia 1-2: 25.37 %.

Para el experimento número tres hemos incluido un patrón totalmente desconocida. Observamos a partir de la imagen resultante que los resultados han sido muy similares a los resultados obtenidos previamente.

4.1.2 Red de Hopfield con el umbral mejorado

La red de Hopfield con el umbral mejorado optimiza el proceso de los resultados del patrón recuperado, esto hace que los elementos del patrón de recuperación no tienda al rango negativo.

4.1.2.1 Experimento 1: Función de biyección



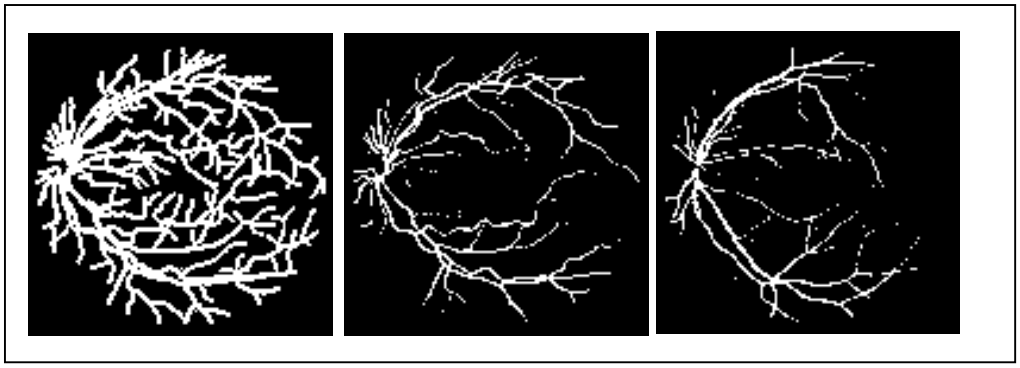
Experimento 1: recuperación de una imagen aprendida en la fase de aprendizaje.

Imagen 1, patrón original; Imagen 2, patrón de salida.

Diferencia 1-2: 0 %.

El número de patrones fundamentales utilizado en este experimento ha sido 6, podemos observar que en este caso la diferencia que hubo entre imagen uno y la imagen dos es cero, es decir, la imagen recuperación ha sido devuelta a su forma inicial

4.1.2.2 Experimento 2: Función de umbral parcial



Experimento 2: recuperación de una imagen con ruido de una imagen aprendida en la fase de aprendizaje.

Imagen 1, patrón aprendido; imagen 2, patrón a recuperar; imagen 3, patrón recuperado.

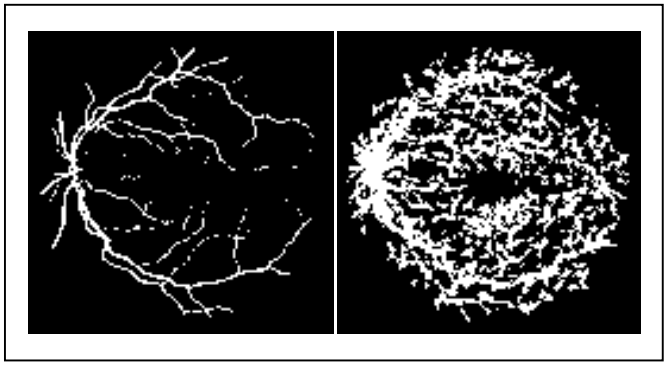
Diferencia 1-2: 19.28 %.

Diferencia 2-3: 23.39 %.

Diferencia 1-3: 27.43 %.

El número de patrones fundamentales utilizado en la fase de aprendizaje ha sido 20. Se ha utilizado una imagen con ruido de más del 20%, donde su imagen inicial ya era aprendida en la fase de aprendizaje. El patrón de salida ha respetado las zonas negras del patrón de recuperación, resultado de la utilización de una función de activación con el umbral parcial.

4.1.2.3 Experimento 3: Función de NPb3



Experimento 3: recuperación de una imagen desconocida por la memoria asociativa.

Imagen 1, patrón a recuperar; imagen 2, patrón recuperado.

Diferencia 1-2: 23.81 %.

El número de patrones fundamentales que se han utilizado en este experimento ha sido 30. Para el experimento número tres hemos incluido un patrón totalmente desconocida. Observamos a partir de la imagen resultante que los resultados han sido muy similares a los resultados obtenidos en la prueba de recuperación de la red de Hopfield (***Apartado 4.1.1.3***), ha simple vista parece ser la misma imagen, pero la diferencia que existe entre ellas ha sido de 1.57%.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

La investigación en este proyecto nos ha permitido conocer mejor los distintos métodos de uso de las redes neuronales, que estas ya no solo se utilizan para calcular la probabilidad y valores de componentes de los patrones, sino que también sirve para recuperar datos informativos que se han perdido anteriormente o simplemente para conocer los nuevos patrones desconocidos. Hemos adaptado dicho algoritmo a nuestra biometría de retinas para su aprendizaje y recuperación transformando las teorías de unos algoritmos en papel a un algoritmo matemático con unos resultados más visuales. Aplicamos estos métodos solo para observar y medir el comportamiento de los algoritmos de entrenamiento. Este nos da un nuevo punto de vista para compararlos y teniendo en cuenta la tolerancia de error en cada recuperación de la nueva neurona. Los resultados que hemos obtenido en el capítulo cuatro no ha podido de considerarse un resultado realmente óptimo ya que en la imagen existe claramente zonas de pixeles no deducibles para el usuario visualizador. Esto puede a deberse la función de activación el umbral escogido. Existen muchas teorías que deducen un umbral con un valor medio entre los dos valores binarios, pero con esa idea hemos llegado a una conclusión de que esa teoría serviría si y solo si el porcentaje y probabilidad de que un pixel tenga valor negro igual que tenga un valor blanco, función de activación cero. Esto no sería posible ya que a primera vista podemos comprobar que los bordes de las imágenes retinas siempre son negras. A pesar de las críticas realizadas por los investigadores anteriores podemos concluir que los resultados obtenidos han sido relativamente aceptables.

5.2 Trabajo futuro

En esta investigación de la biometría de las retinas hemos utilizado este algoritmo de memoria asociativa ya que consideramos que es la herramienta perfecta. Ésta técnica de aprendizaje y recuperación ya no se utiliza para este tipo de patrones. Podríamos utilizar dicho algoritmo también para análisis de imágenes fotográficas, huellas dactilares, palmas de las manos, etc. Las existencias de estos algoritmos únicamente tendrían valores invalorables si tuvieran una función global a todos los problemas planteados, y esta es la labor de los futuros investigadores.

Referencias

1. James E. Anderson (1995) “An indroduction to Neural Networks, MIT Press” http://www.nbb.cornell.edu/neurobio/linster/lecture3.pdf
2. Redes Neuronales Artificial http://electronica.com.mx/neural/informacion/hopfield.html
3. Inteligencia Artificial. s.f. https://es.wikipedia.org/wiki/Inteligencia_artificial.
4. Modelo de Hopfield http://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo5.html
5. Red de Hopfield http://magomar.webs.upv.es/rna/tutorial/RNA_hopfield.html
6. Red neuronal artificial. s.f. https://es.wikipedia.org/wiki/Red_neuronal_artificial.
7. Associative Memories. s.f. http://wing.comp.nus.edu.sg/pris/AssociativeMemory/LinearAssociator.htm

Glosario

API	Application Programming Interface
RNA	Redes neuronales artificiales.
ANN	Artificial neural network
ART	Adaptative Resonance Theory

Anexos

Código Matlab: procesamiento de imágenes

(i) Binarización de las imágenes

```
function imagen = Binarizacion(fichero,umbral,longitud)
    imaFile = strcat(fichero,'.gif');
    textFile = strcat(fichero,'.txt');

    im = imread (imaFile,'gif');
    SE = strel('ball',1,1);
    Dilate = imdilate(im,SE);
    imD = imresize(double(Dilate),[longitud,longitud]);
    imD = im2bw(imD);

    fp = fopen(textFile, 'w+');
    for i = 1:longitud
        for j = 1:longitud
            if imD(i,j) <= umbral
                fprintf(fp,'-1 ');
            else
                fprintf(fp,'1 ');
            end
        end
    end

    fclose(fp);
    imwrite(imD,imaFile,'gif');
    imagen = imD;
```

(ii) Valores binarios a imagen

```
function image = vec2ima(fileName,imSize)
    name = strcat(fileName,'.txt');
    imName = strcat(fileName,'.bmp');

    fp = fopen(name,'r');
    vec = fscanf(fp,'%d');
    mt = vec2mat(vec,imSize);
    for row = 1:imSize
        for col = 1:imSize
            if mt(row,col) < 0
                mt(row,col) = 0;
            else
                mt(row,col) = 255;
            end
        end
    end

%     imshow(mt);
    imwrite(mt,imName,'bmp');
    image = uint8(mt);
```

Código C: arquitectura del sistema

(i) Infodato.h

```
// Mostrar informacion de salida y en fichero.
extern BOOL printOut;
extern BOOL printLog;
extern FILE *fpLog;
extern char msg[128];
extern int contTraining;

void MensajeDeSalida(char *msg);

Patron *inicializarPatron();
void liberarPatron(Patron *patron);

Peso *inicializarPeso();
void liberarPeso(Peso *peso, int tamanio);

Elemento *inicializarElemento();
void liberarElemento(Elemento *elemento);
Elemento *insertarElemento(char *fichero);
void SumPixelElemento(Elemento *elemento);
float PorcentajeBlanco(Elemento *elemento);
float PorcentajeNegro(Elemento *elemento);

Lista *inicializarLista();
void liberarLista(Lista *lista);
STATUS insertarLista(Lista *lista, Elemento *elemento);
STATUS extraerLista(Lista *lista, int posicion);

STATUS importarPatrones(Lista *lst, char *directorio);
STATUS exportarPatrones(Lista *lst, char *directorio);
```

(ii) Calculardato.h

```
int DistanciaEntreElementos(Elemento *e1, Elemento *e2);
int DistanciaEntreElementosDiferencia(Elemento *e1, Elemento *e2);
STATUS SumatorioPatron(Elemento *elemento);

double MediaAritmetica(Elemento *elemento);
double MediaAritmeticaConjunto(Lista *lista);

STATUS FaseDeAprendizaje(Elemento *elemento);
STATUS PesoHebbianoProgresado(Elemento *elemento, int tiempo, double noerror);
STATUS FaseDeAprendizajeConjuntoPre(Lista *lista);
STATUS FaseDeAprendizajeConjuntoPost(Lista *lista);
STATUS ActualizarFaseDeAprendizaje(Lista *lista, Elemento *elemento);

STATUS FaseDeRecuperacion(Lista *lista, Elemento *test, double umbral);

STATUS importarPatronesPorBloque(Lista *lst, char *directorio);

double UmbralSigmoidal(Lista *lista, float cnt);
```

(iii) Distancia de euclides de dos vectores

```
int DistanciaEntreElementos(Elemento *e1, Elemento *e2) {

    int i, distancia, maxposisiones;

    if ((e1 == NULL) || (e2 == NULL))
        return -999999999;

    distancia = 0;
    maxposisiones = MATRIZ_X * MATRIZ_Y;
    for (i = 0; i < maxposisiones; i++)
        distancia += abs(e1->dato->pixel[i] - e2->dato->pixel[i]);

    sprintf(msg, "DistanciaEntreElementos(): (%p<->%p) %d", e1, e2, distancia);
    MensajeDeSalida(msg);
    return distancia;
}
```

(iv) Distancia de vectores por diferencia

```
int DistanciaEntreElementosDiferencia(Elemento *e1, Elemento *e2) {

    int i, maxposisiones;
    float distancia;

    if ((e1 == NULL) || (e2 == NULL))
        return -999999999;

    distancia = 0;
    maxposisiones = MATRIZ_X * MATRIZ_Y;
    for (i = 0; i < maxposisiones; i++)
        if (e1->dato->pixel[i] != e2->dato->pixel[i])
            distancia = distancia + 1;

    distancia = (distancia * 100) / (MATRIZ_X*MATRIZ_Y);
    sprintf(msg, "DistanciaEntreElementosDiferencia(): (%p<->%p) %.2lf %%", e1, e2, distancia);
    MensajeDeSalida(msg);
    return distancia;
}
```

(v) Fase de aprendizaje

```
STATUS FaseDeAprendizaje(Elemento *elemento) {

    int i, j, maxposiciones;

    if (elemento == NULL) return ERR;
    if (elemento->dato == NULL) return ERR;

    if (elemento->pesow == NULL) {
        elemento->pesow = inicializarPeso();
        if (elemento->pesow == NULL)
            return ERR;
    }

    maxposiciones = MATRIZ_X * MATRIZ_Y;
#pragma omp parallel for default(none) \
        private(i, j, auxiliar) \
        shared(maxposiciones, elemento, distancia, existe)
    // matriz resultante
    for (i = 0; i < maxposiciones; i++) {
        for (j = 0; j < maxposiciones; j++) {
            if (i != j)
                elemento->pesow->matriz[i][j] =
                elemento->dato->pixel[i] *
                elemento->dato->pixel[j];
            else
                elemento->pesow->matriz[i][j] = 0;
        }
    }

    sprintf(msg, "FaseDeAprendizaje(): OK");
    MensajeDeSalida(msg);
    return OK;
}
```

(vi) Actualización de pesos en la fase de aprendizaje

```
STATUS ActualizarFaseDeAprendizaje(Lista *lista, Elemento *elemento) {

    int i, j, maxposiciones;

    if ((lista == NULL) || (elemento == NULL))
        return ERR;

    if (elemento->dato == NULL) return ERR;

    maxposiciones = MATRIZ_X * MATRIZ_Y;
    if (lista->pesoConjunto == NULL) {
        lista->pesoConjunto = inicializarPeso();
        if (lista->pesoConjunto == NULL)
            return ERR;
        for (i = 0; i < maxposiciones; i++)
            for (j = 0; j < maxposiciones; j++)
                lista->pesoConjunto->matriz[i][j] = 0;
    }

    if (elemento->pesow == NULL)
        if (FaseDeAprendizaje(elemento) == ERR)
            return ERR;

    for (i = 0; i < maxposiciones; i++)
        for (j = 0; j < maxposiciones; j++)
            lista->pesoConjunto->matriz[i][j] += elemento->pesow->matriz[i][j];

    return OK;
}
```

(vii) Fase de recuperación

```
STATUS FaseDeRecuperacion(Lista *lista, Elemento *test, double umbral) {

    int res_i, peso_j, maxposiciones, suma, sumatorio;

    if ((lista == NULL) || (test == NULL))
        return ERR;

    if (lista->pesoConjunto == NULL)
        return ERR;

    maxposiciones = (MATRIZ_X * MATRIZ_Y);
    suma = 0;
    for (res_i = 0; res_i < maxposiciones; res_i++) {
        sumatorio = 0;
        for (peso_j = 0; peso_j < maxposiciones; peso_j++) {
            sumatorio += test->dato->pixel[peso_j] *
                lista->pesoConjunto->matriz[res_i][peso_j];
        }

        if (sumatorio > umbral) {
            test->dato->pixel[res_i] = 1;
        }
        else if (sumatorio < umbral) {
            test->dato->pixel[res_i] = -1;
        }
        else {
            printf("%d", test->dato->pixel[res_i]);
        }

        suma += test->dato->pixel[res_i];
    }

    test->dato->SumPixeles = suma;
    sprintf(msg, "FaseDeRecuperacion(): OK");
    MensajeDeSalida(msg);
    return OK;
}
```

(viii) Calcular función de umbral

```
double UmbralSigmoidal(Lista *lista, float cnt) {

    double umbral, media;

    if (lista == NULL) return ERR;

    if ((media = MediaAritmeticaConjunto(lista)) == ERR)
        return ERR;

    media = media * (1 - media);
    umbral = cnt * (MATRIZ_X * MATRIZ_Y) * pow(media, 3) * lista->cantidad;
    sprintf(msg, "UmbralSigmoidal(): %.2lf const: %.3lf", umbral, cnt);
    MensajeDeSalida(msg);
    return umbral;
}
```